NSY104

Architectures des systèmes informatiques

2010-2011

TD N°3 – Mémoire cache

1. **Alignement**

Pour un cache dont les lignes font 128 octets, donnez l'adresse du premier mot dans la ligne contenant l'adresse suivante :

– 0xA23847EF

– 0x7245E824

– 0xEEFABCD2

Correction : *Si les lignes du cache font 128 octets, alors les 7 bits de poids faible de l'adresse indiquent à quel octet dans la ligne une adresse fait référence. Les lignes étant alignées, l'adresse du premier mot dans la ligne peut être trouvée en positionnant à 0 les bits spécifiant les octets dans la ligne. En conséquence, les adresses des premiers octets dans les lignes contenant les adresses précitées sont les suivantes :*

– 0xA2384780

– 0x7245E800

– 0xEEFABC80

2. **Lignes et longueurs de lignes**

Soit un cache d'une capacité de 32 Ko. Combien de lignes peut contenir le cache si les longueurs de ligne sont de 32, 64 ou 128 octets ?

Correction : *Le nombre de lignes d'un cache correspond simplement à sa capacité divisée par la longueur de ligne, aussi le cache possède-t-il 1024 lignes si les lignes font 32 octets, 512 si elles font 64 octets et 256 si elles en font 128.*

3. **Associativité et ensembles**

Si un cache possède une capacité de 16 Ko et une longueur de ligne de 128 octets, combien d'ensembles le cache possède-t-il s'il est associatif par ensemble de 2, 4 ou 8 blocs ?

Correction : *Avec des lignes de 128 octets, le cache contient un total de 128 lignes. Le nombre d'ensembles dans le cache correspond au nombre de lignes divisé par le degré d'associativité du cache. Le cache possède donc 64 ensembles pour une associativité par ensembles de 2 blocs, 32 ensembles pour une associativité par ensemble de 4 blocs et 16 ensembles pour une associativité par ensemble de 8 blocs.*

4. **Taille des tableaux d'étiquettes**

Un cache possède une capacité de 64 Ko, des lignes de 128 octets et un degré d'associativité de 4. Le système contenant le cache utilise des adresses de 32 bits.

– Combien de lignes et d'ensembles possède le cache ?

Correction : *64 Ko/128 octets = 512. Le cache possède donc 512 lignes. Puisque son degré d'associativité est de 4, le cache possède 512/4 = 128 ensembles.*

– Combien d'entrées sont requises dans le tableau d'étiquettes ?

Correction : *Une entrée de tableau d'étiquette est requise pour chaque ligne, aussi le tableau d'étiquettes nécessite-t-il 512 entrées.*

– Combien de bits d'étiquettes sont requis pour chaque entrée dans le tableau d'étiquettes ?

Correction : *Puisque le cache possède 128 ensembles, 7 bits d'adresse seront utilisés pour sélectionner un ensemble dans le tableau, 7 bits supplémentaires seront utilisés pour sélectionner l'octet à l'intérieur de chaque ligne, car les lignes font 128 octets. En conséquence, 18 (32-14) bits d'étiquette sont requis pour chaque entrée dans le tableau d'étiquette.*

– Si le cache est de type write-through, combien de bits sont requis pour chaque entrée du tableau d'étiquettes et quelle quantité de mémoire totale est requise pour le tableau dans le cas d'une politique de remplacement LRU? Qu'en serait-il s'il s'agissait d'un cache write-back ?

Correction : *Puisque le cache utilise la politique de remplacement LRU et possède un degré d'associativité de 4, 2 bits sont requis pour chaque entrée du tableau d'étiquettes pour mémoriser l'ordre du dernier référencement. Les*

caches write-through nécessitent un bit de validité dans chaque entrée du tableau du tableau d'étiquette mais pas de bit de modification, ce qui nous donne une taille de 21 bits (18+2+1) pour chaque entrée du tableau d'étiquettes. Puisqu'il y a 512 lignes dans le cache, la taille du tableau d'étiquettes sera de 10 752 bits. Les caches write-back nécessitent un bit de modification dans chaque entrée du tableau d'étiquettes en plus des bits requis pour un cache write-through similaire, aussi chaque entrée du tableau d'étiquettes requerra 22 bits, pour une taille totale du tableau de 11 264 bits.

5. **Adressage**

Soit un cache possédant les mêmes caractéristiques que celui de l'exercice précédent. Pour chacune des adresses mentionnées ci-après, indiquez le numéro de l'ensemble qui sera examiné afin de déterminer si l'adresse est contenue dans le cache et celui de l'octet référencé dans la ligne de cache. Nous supposerons que les bits utilisés pour sélectionner un octet à l'intérieur de la ligne sont les bits de poids faible de l'adresse et que les bits utilisés pour sélectionner l'ensemble sont les bits suivants de poids plus élevé.

– 0xABC89987
– 0x32651987
– 0x228945DB
– 0x48569CAC

Correction : *Dans l'exercice précédent, nous avons conclu que 7 bits étaient requis pour sélectionner un octet dans la ligne de cache, aussi les 7 bits de poids faible de l'adresse détermineront l'octet de référence. Sept bits sont également requis pour sélectionner l'ensemble, qui correspondra aux bits 7 à 13 de l'adresse. Considérant cela, les réponses sont les suivantes :*

– 0xABC89987
Octet dans la ligne = 0x7, ensemble = 0x33 (51).
– 0x32651987
Les 14 bits de poids faible de cette adresse sont les mêmes que ceux de la question précédente, aussi, l'octet et l'ensemble seront-ils les mêmes.
– 0x228945DB
Octet dans la ligne = 0x5B (91), ensemble = 0xB (11).
– 0x48569CAC
Octet dans la ligne = 0x2C (44), ensemble = 0x39 (57).

6. **Taux de hit et temps d'accès**

Supposons qu'un cache possède un temps d'accès (latence de cache-hit) de 10 ns et un taux de miss de 5 %. Une modification apportée au cache ferait baisser son taux de miss à 3 % mais ferait monter la latence du cache-hit à 15 ns . Dans quelles conditions cette modification pourrait-elle offrir de meilleures performances (temps d'accès moyen à la mémoire plus court) ?

Correction : *Le temps d'accès moyen à la mémoire est* $(T_{hit} \times P_{hit}) + (T_{miss} \times P_{miss})$. *Pour que la modification permette de réduire le temps d'accès, il est nécessaire que* $(15\ ns \times 0{,}97) + (T_{miss} \times 0{,}03) < (10\ ns \times 0{,}95) + (T_{miss} \times 0{,}05)$. *En résolvant cette inégalité, on obtient* $T_{miss} > 252{,}5\ ns$. *Dans les cas où le temps de cache-miss est supérieur à cette valeur, la réduction de fréquence des cache-miss aura plus d'impact sur le temps d'accès moyen que l'augmentation du temps de cache-hit. De manière intuitive, ce résultat paraît logique : à mesure que le temps de cache-miss devient plus important, nous serons plus facilement prédisposés à augmenter le temps de cache-hit pour réduire le taux de cache-miss, parce que chaque cache-miss devient de plus en plus pénalisant.*

7. **Taux de hit et temps d'extraction de ligne**

Un cache possède un taux de hit de 95 %, des lignes de 128 octets et une latence de cache-hit de 5 ns. La mémoire principale prend 100 ns pour retourner le premier mot (32 bits) d'une ligne puis 10 ns pour retourner chaque mot suivant.

– Quelle est la valeur de T_{miss} pour ce cache ? Nous supposerons que le cache attend que la ligne soit chargée dans le cache et réexécute ensuite l'opération mémoire en obtenant alors un cache-hit. Nous négligerons le temps requis pour écrire la ligne dans le cache une fois qu'elle a été extraite de la mémoire principale. Nous supposerons également que le cache prend le même temps pour détecter un miss qu'il n'en requiert pour un cache-hit.

Correction : *Le cache possède des lignes de 128 octets qui correspondent à 32 mots. En considérant les temps d'opération mémoire indiqués, le cache nécessite 5 ns pour détecter qu'un miss est survenu et* $100\ ns + (31 \times 10\ ns) = 410\ ns$ *pour extraire une ligne de la mémoire principale et la placer dans le cache. Le cache réexécute alors l'opération qui a causé le miss, ce qui prend 5 ns. Ceci donne un temps de cache-miss de 420 ns.*

– Si le fait de doubler la longueur de ligne du cache permet de réduire le taux de miss à 3 %, le temps d'accès moyen à la mémoire s'en trouvera-t-il réduit ?

Correction : *Pour le cache de départ, le temps d'accès moyen à la mémoire est* (0,95×5 *ns*)+(0,05× 420 *ns*) = 25,75 n*s. Si nous doublons la longueur de ligne, les lignes de cache font 64 mots et nécessitent 730 ns pour être extraites de la mémoire, aussi la latence du cache miss atteint-elle 740 ns. Le temps moyen d'accès à la mémoire devient* (0,97×5 *ns*)+(0,03×740 *ns*) = 27,05 *ns. Si nous opérions ce changement, le temps moyen d'accès à la mémoire serait donc au contraire augmenté.*

8. **Échecs obligatoires, de capacité et de conflit**

Un programme accède à 1 000 000 de références mémoire. Lorsqu'il tourne sur un système donné, le cache obtient un taux de miss de 7 %, dont un quart sont des échecs obligatoires, un autre quart des échecs de capacité et la moitié des échecs de conflits.

– Si la seule modification que vous pouvez apporter au cache consiste à augmenter son degré d'associativité, quel est le nombre maximal de miss que vous pouvez espérer éliminer ?

Correction : *Le fait d'augmenter le degré d'associativité permettra de réduire le nombre d'échecs de conflits (des miss qui surviennent parce que les lignes de cache se font concurrence pour des emplacements dans le cache), mais n'affectera pas le nombre d'échecs de capacité (les miss qui surviennent parce qu'un programme référence plus de données que ne peut en contenir le cache). En conséquence, le mieux que nous puissions espérer en augmentant le degré d'associativité du cache est d'éliminer tous les échecs de conflit. Le taux de miss étant de 7 % et le programme opérant 1 000 000 de références mémoire, le nombre total de miss est de 70 000. La moitié d'entre eux sont des échecs de conflit. Le nombre maximale que nous pouvons éliminer en augmentant le degré d'associativité du cache est donc de 35 000.*

– Si vous avez la possibilité d'augmenter la taille du cache en plus de son degré d'associativité, quel est le nombre maximal de miss que vous pouvez espérer éliminer ?

Correction : *En augmentant la capacité ainsi que le degré d'associativité du cache, nous pouvons éliminer les miss de capacité et les miss de conflit à la fois, soit les trois quarts de l'ensemble des miss, pour un total de 52 500 miss.*